IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of:	Cary L. Bates, et al.	:	Date: March 8, 2007
Group Art Unit:	2192	:	IBM Corporation
Examiner:	J. Romano	:	Intellectual Property Law
Serial No.:	10/008,864	:	Dept. 917, Bldg. 006-1
Filed:	December 6, 2001	:	3605 Highway 52 North
Title: STORING AND RESTORING SNAPSHOTS OF A COMPUTER PROCESS		:	Rochester, MN 55901

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 223313-1450

APPEAL BRIEF IN SUPPORT OF APPEAL FROM THE PRIMARY EXAMINER TO THE BOARD OF APPEALS

Sir:

This is an appeal of a Final Rejection under 35 U.S.C. §103(a) of claims 1, 2, 4, 5, 7-12 and 25-32 of Application Serial No. 10/008,864, filed December 6, 2001. This brief is submitted pursuant to a Notice of Appeal filed January 8, 2007, as required by 37 C.F.R. §1.192.

1. Real Party in Interest

International Business Machines Corporation of Armonk, NY, is the real party in interest. The inventors assigned their interest as recorded on December 6, 2001, on Reel 012379, Frame 0410.

2. Related Appeals and Interferences

There are no related appeals nor interferences pending with this application.

3. Status of Claims

Claims 1, 2, 4, 5, 7-12 and 25-32 are pending and stand finally rejected. The claims on appeal are set forth in the Appendix of Claims

4. Status of Amendments

No amendments were submitted following Final Rejection.

5. Summary of Claimed Subject Matter

The invention herein relates to the use of automated tracing and debugging tools for analyzing and/or debugging computer programming code. Independent claims 1 and 12 recite methods of tracing, independent claim 25 recites an article of manufacture in the form of a data storage medium embodying a program for tracing (i.e, a computer program product), and independent claim 28 recites a data processing device which performs tracing.

A user specifies a computer process and a trigger expression to be traced during execution of the process (Spec. p. 4, line 4-6; p. 11, lines 21-26; p. 15, lines 6-19; p. 16, lines 3-16; Figs. 5 & 6B). The trigger expression is a non-executable data value having a state (as opposed to program instructions), per all independent claims, and in particular is an "L-value", meaning a value that can be expressed as being left of an equal sign in a logical or arithmetic expression, per independent claim 12 (Spec. p. 12, lines 13-18). A trace tool then monitors execution of the specified process for occurrences of references to the specified trigger expression (Spec p. 11, lines 25-29). When the specified trigger expression is referenced, certain state data is stored (Spec. p. 11, lines 25-29). Storage of state data upon referencing the specified trigger expression can be contingent on some additional condition, per independent claim 12 (Spec. p. 4, lines 10-11; p. 15, lines 24-25; Fig. 5 feature 526). State data (referred to as a "snapshot") includes the state of the trigger expression, and may include additional state data as well (Spec. p. 4, lines 11-23; p. 11, lines 26- p. 12, line 6; Figs. 3A and 3B). The state data can later be displayed (per independent claims 12, 25 and 28) and/or restored (per independent claims 1, 12 ans 25) (Spec. p. 13, line 20 - p. 15, line 5; Fig. 4).

6. Grounds of Rejection To Be Reviewed on Appeal

Claims 1, 4, 5, and 7-11 are finally rejected under 35 U.S.C. §103(a) as unpatentable over Wygodny et al. (US 6,282,701) in view of Matt et al.(US 6,510,507). Claims 2, 12 and 25-32 are rejected under 35 U.S.C. §103(a) as unpatentable over *Wygodny* and *Matt*, further in view of Lindsey (US 5,96,536). The only issues in this appeal are whether the claims are prima facie obvious over *Wygodny*, *Matt* and *Lindsey*.

7. Argument

Appellants contend that the Examiner failed to establish adequate grounds of rejection for the following reasons:

I. The Examiner improperly rejected the claims under 35 U.S.C. §103(a) because neither *Wygodny* nor *Matt* (nor *Lindsey*), considered alone or in combination, discloses the key features of appellant's independent claims, i.e., receiving a user specification of a "trigger expression", and responsive thereto collecting trace data upon referencing the trigger expression during execution. [page 7 below]

II. The Examiner improperly rejected the claims under 35 U.S.C. §103(a) because the devices disclosed in *Wygodny* and *Matt* are intended for different environments, and a suggestion to combine the references is lacking. [page 12 below].

Overview of Invention

A brief overview of appellants' invention in light of existing art will be helpful in appreciating the issues herein. ***Appellants' invention is intended to improve an interface used for tracing and debugging computer programs***. Trace techniques are well known in the art, and have been used to gather data generated during the execution of a computer program in order to analyze the behavior of that computer program, identify the source of errors, and so forth. Conventionally, a programmer wishing to trace program execution specifies one or more tracepoints (or breakpoints) in the code. A tracepoint is a location in the sequence of code statements. In some embodiments, a trace statement is inserted into the stream of source code statements, which becomes compiled as an executable code statement causing certain trace data to be collected. In other embodiments, a trace mechanism monitors the instruction stream during execution for occurrences of a specified instruction location. In either case, what triggers the collection of trace data is encountering a location in the instruction stream. Upon the particular location in the instruction stream being encountered, state data as specified by the user is collected as trace data. It is further known to specify additional conditions on the collection of trace data, e.g., trace data will be collected at a particular instruction stream location only if some additional condition, such as a state variable being equal to a specified value, is met.

A recurrent problem in collecting trace data, as it is in analyzing program execution behavior in general, is data overload. I.e., a programmer can very easily generate an enormous amount of data, but finding the data of interest in this mountain of trace data can be very tedious.

Appellants' invention is intended to address one specific aspect of this problem of data overload. It does not necessarily provide a "magic bullet" which solves all aspects of computer program analysis, but provides a useful tool for extracting data of interest in certain circumstances.

Appellants observed that in certain programming environments, the same code sequences are re-executed many times for different data expressions. A programmer may be interested in the behavior of a single data expression, out of multiple such data expressions. A simple example would be a single array element from a large array. Using conventional techniques, it is possible to specify relevant code statements which might access the data of interest (and might be used access other data), to collect a massive amount of trace data, and to cull out the references to other data expressions which are not of interest. This is obviously very tedious for the programmer. Appellants therefore propose to provide a tool for use in such cases, in which the programmer ***does not specify a code location to be monitored, but specifies a data expression instead.***

This data expression of interest is referred to as a "trigger expression". Upon a reference being made to the data expression during execution, appropriate trace data is collected. Thus, if the same code sequence is used to access many different data expressions, tracing will be triggered only for a relatively small subset of such accesses, which the programmer has indicated are of interest.

As so often happens in the case of inventions in the realm of improved user interface, appellants' invention is intended to make life easier for the programmer, and not to collect data which would have been impossible to obtain using conventional techniques. Appellants make no assertion that their invention provides a capability to collect data which was heretofore impossible to collect. It has long been possible to collect all possible data regarding a specific data expression by identifying each and every code sequence which might access that data, and triggering trace data collection whenever such a code sequence is encountered. As explained above, the problem with that approach is that, in some circumstances, it will collect a massive amount of irrelevant data in addition to the data of interest. It is further possible using conventional techniques to reduce the amount of data by further specifying conditions on data collection at the trace points, but this again requires further effort by the programmer, and runs the risk of error by specifying conditions which might be too restrictive or not sufficiently restrictive. Appellants' invention makes it easier for a programmer to analyze and debug computer programming code in certain circumstances by allowing the programmer to directly specify a data expression to be monitored, thereby reducing the amount of irrelevant data collected and/or effort to specify and cull data..

I. The Examiner improperly rejected the claims under 35 U.S.C. §103(a) because neither *Wygodny* nor *Matt* (nor *Lindsey*), considered alone or in combination, discloses the key features of appellant's independent claims, i.e., receiving a user specification of a "trigger expression", and responsive thereto collecting trace data upon referencing the trigger expression during execution.

In order to support a rejection for obviousness, there must be some suggestion in the art to combine the references in such a manner as to form each and every element of appellants' claimed invention. It is not sufficient that a suggestion may exist to combine the references, if such a combination does not meet the limitations of appellants' claims without some further non-obvious modification. Appellants will, for the moment, defer the issue of whether it was proper to combine the references. Even assuming arguendo that a suitable suggestion exists in the art to combine the references, the hypothetical combination fails to teach or suggest the significant claimed features of appellants' invention, and therefore the rejection was improper.

Wygodny, the primary reference cited herein, discloses a tracing system for remote use, in which trace files can be generated during program execution for later analysis at a remote location. Although *Wygodny* refers to "tracing" values of state variables, they are referring to the ***data that is collected with the trace, not the event which triggers the collection of data***. I.e., once a triggering event is encountered, the values of specified state variables can be saved. *Wygodny* discloses that the ***triggering event*** which causes trace data to be collected is encountering one or more trace points ***in the instruction stream of the code*** during execution. *Wygodny* discloses that the user specifies these trace points, i.e. locations in the instruction stream of the code, before execution, and upon being encountered during execution, the values of the appropriate state variables are saved. Thus, *Wygodny* is essentially a conventional trace technique which has been adapted for remote use.

Matt discloses a hardware mechanism for comparing addresses on a data communications bus to determine whether a bus address matches a pre-determined address or range of addresses to be monitored. Specifically, *Matt*'s hardware mechanism uses a set of hardware reference registers to match an upper portion of a bus address and a separate page table which is indexed by the lower address bits to determine a match of the lower portion. Such a mechanism supports rapid address matching for an arbitrary range of addresses within a page. The mechanism can be used to monitor access to particular memory locations. Specifically, it is used in an environment in which a debug host computer (such as a PC) monitors a "target system" containing one or more processor cores through a special hardware debug interface.

Matt specifically discloses that a user of the debug host computer can control execution of the target system by using typical commands such as run, halt and step, or by specifying "debug events", such as program counter breakpoints, or "watchpoints". A "watchpoint" is a memory reference (i.e. memory address or range of memory addresses) to be monitored, which cause execution to halt.

The Examiner reasons that since *Matt* discloses the capability to monitor specific memory addresses as "debug events" which cause a halt to program execution, it would have been obvious to combine this capability with conventional software trace technique disclosed in *Wygodny* to find the recited elements of appellants' invention.

The problem with this line of reasoning is that it is abstracting appellants' invention to a high level, e.g., 'triggering trace collection on a memory reference', without considering the specific limitations of the claims.

Appellants' representative claim 1 recites:

1. A method of tracing the activity of an expression, said method comprising the machine-implemented steps of:

(a) receiving, from a user, a specification of a machine-implemented process in which a trigger expression is to be traced;

(b) *receiving, from a user, a specification of the trigger expression to be traced in the machine-implemented process, said trigger expression representing a non-executable data value having a state*;

(c) responsive to steps (a) and (b), monitoring execution of said machine-implemented process to detect occurrences of a plurality of references to a location in machine memory representing a state of said trigger expression, wherein each said occurrence of a *reference to a location in machine memory representing a state of said trigger expression occurs as a result of executing said machine-implemented process*;

(d) responsive to each detected occurrence of a reference to said location in machine memory representing a state of said trigger expression, storing the respective state of the trigger expression at the time of the respective detected occurrence of a reference to said location in machine memory representing a state of said trigger expression to create a history of said trigger expression within the machine-implemented process, said storing step being performed without interrupting the machine-implemented process; and

(e) restoring the state of the trigger expression when requested.

[emphasis added]

The remaining independent claims, while varying in scope, all contain analogous limitations to the italicized language above. Independent claim 12 (as well as dependent claim 5) further recite that the trigger expression results in an L-value during execution[1]

What the claims recite is that the ***user specifies a "trigger expression"***. It is true that ultimately the state of the trigger expression is stored in some memory location, and that at a low level one monitors the trigger expression during execution by monitoring

[1] An "L-value" is defined in the Specification as "a value that can be expressed as being left of an equal sign in a logical or arithmetic expression". Examples include an array element, a pointer expression, a substring expression, etc., as well as (named) variables. (Spec. p. 12, lines 16-18)

references to the memory location at which it is stored. But neither *Matt* nor *Wygodny* teaches that the user specifies a trigger expression to be monitored. *Wygodny* discloses that the user specifies a ***code statement*** to trigger collection of data, but a code statement is clearly not a "non-executable data value having a state", as recited in the claims.[2] *Matt* discloses a low-level hardware monitoring mechanism, and does not disclose any particular form of user interface. However, to the extent *Matt* suggests any particular user input, the implication is that the user specifies a ***memory location*** or range of memory locations to be monitored.

Appellant is compelled to point out that the anticipated environment for use of *Matt*'s disclosed device is somewhat different, i.e., *Matt*'s invention is a hardware monitor which is apparently used for low-level hardware debug, requiring a special low-level hardware interface. In such an environment, the capability to monitor particular address ranges may be desirable. A high-level variable expression is not the same as a memory location, even if the variable's state is stored at a memory location. The capability to monitor a bus address is not tantamount to a suggestion that a user specifies a trigger expression at a high level to be monitored.

One must realistically ask, just what does the hypothetical combination of *Wygodny* and *Matt* teach or suggest? *Wygodny* indeed discloses a mechanism for tracing execution of software processes, but the only triggering event disclosed is to trigger on previously specified code statements. *Matt* discloses a low level hardware monitor with the capability to detect particular memory range references. The combination arguably

[2] The Examiner originally read "trigger expression" to encompass a code statement and thus found the claims anticipated by *Wygodny*, but appears to have abandoned that position in view of clarifying amendments made by appellants.

suggests that one could use a low-level hardware mechanism of *Matt* to monitor triggering events as specified by *Wygodny*. It might even be said to suggest that one could specify a memory location to be traced. But none of these things amounts to a suggestion of the recited features of appellant's invention. Specifically, none of these things suggest that the user specify a "trigger expression".

Although appellants' recited "trigger expression" is ultimately stored at a memory location, specifying a memory location by the user is substantially different from specifying a trigger expression. Typically, the "trigger expression" has no fixed memory location, but is translated to a memory location as required. The user debugging a high-level process is not interested in the memory location, but in the "trigger expression. Thus, neither *Wygodny* nor *Matt*, considered alone or in combination, teaches or suggests that a user specifies a "trigger expression" to be monitored, as that phrase is used in appellants' claims.

Independent claim 12 and dependent claim 5 further recite that the trigger expression results in an L-value during execution of the program being traced, and thus further clarify, to the extent such clarification is necessary, the nature of the trigger expression. For reasons explained above this limitation is similarly not met by *Wygodny* and *Matt*, either alone or in combination.

Lindsey, the tertiary reference, discloses a technique for tracing certain activity with respect to object data in an object-oriented programming language. According to Lindsey, messages intended for a specified data object instance in an object-oriented program are intercepted, and trace data is captured depending on the parameters of the message. *Lindsey* is cited to show imposing conditions on the triggering of trace data collection. I.e., according to *Lindsey*, conditions can be attached to the collection of trace data from an intercepted message. *Lindsey* does not teach or suggest, alone or in combination with *Wygodny* and *Matt*, the essential limitations with respect to a triggering expression discussed above.

For all the reasons explained above, the combination of the cited references (even assuming it was proper to combine the references fails to teach or suggest the critical limitations of appellants' claims, and the rejection of the claims was erroneous.

II. The Examiner improperly rejected the claims under 35 U.S.C. §103(a) because the devices disclosed in *Wygodny* and *Matt* are intended for different environments, and a suggestion to combine the references is lacking.

In the previous discussion, appellants granted, for the sake of argument, that it was proper to combine *Wygodny* and *Matt*. But in fact such a combination would not have been suggested by the references or the art, and the hypothetical combination, even if it did contain all the recited elements of appellants' claims, was improper.

On a superficial view, it may seem that the combination was proper. After all, both *Wygodny* and *Matt* relate is a general way to trace techniques. But there is a subtle difference. *Wygodny* relates to tracing of software processes at a higher level, in which the user conventionally specifies tracepoints and breakpoints in the form of code statements which cause some action to be taken. *Matt* relates to tracing of processor cores or similar hardware components, in which the operation of the components is traced by observing the contents of registers, buses and so forth.

The probable retort that appellants' claims do not recite high-level or low-level processes misses the point. Appellants' claims do in fact recite that the user specifies a "trigger expression" and/or "L-value", which are high-level language constructs. The ultimate issue is whether specifying a "trigger expression", i.e. a high level language construct, is suggested by the references. There would be little reason to specify a "trigger expression" in a low-level hardware interface trace tool such as *Matt*'s, because the purpose of such a tool is to directly diagnose hardware operation, and by-pass many of the operating system constructs which support high-level languages.

As a practical matter, a user in a high-level software tracing process such as *Wygodny*'s and appellants' tracing processes does not specify a memory location to be traced (as in *Matt*) for the simple reason that ***the user can't possibly know the memory location***. Generally, application programming code of the type contemplated in *Wygodny* (and by appellants), i.e. high-level language code, is relocatable. As memory is paged in and out during execution, a particular variable may be assigned to different real memory addresses. When *Wygodny*'s user specifies a code statement to be traced, he does not specify the real memory address at which the code statement is to be stored. Again, he can not possibly know that real memory address. It is not determined until execution time, and even then it may change multiple times during execution.

Matt's capability to monitor a particular memory address on a hardware bus is obviously intended for hardware debug at a very low level, and not for the type of software tracing disclosed in *Wygodny* and appellants' specification. For all of these reasons, there is no suggestion in the art or in the references themselves to combine them as done by the Examiner, and the Examiner's rejections were improper.

8. Summary

Appellant discloses and claims a novel and unobvious technique for tracing software processes, in which a user specifies a "trigger expression", the trigger expression representing a data value having a state, and selective trace data is collected during execution upon referencing a memory location storing the state of the trigger expression. This invention makes it easier in certain circumstances for a user to collect a limited amount of data relating to a specific software variable. While it is possible to collect similar data using prior art techniques of specifying code statements to be traced and optional conditions for tracing, it is generally more burdensome and/or error prone. The primary reference, *Wygodny*, discloses a conventional tracing technique which specifies code statements to be traced. The secondary reference, *Matt*, discloses hardware tracing at a low level in which it is possible to trace particular addresses on a bus. None of th references, alone or in combination, teaches or suggests the specification of a "trigger expression" for direct tracing, as recited in appellants' claims, and the combination of *Wygodny* and *Matt* was, in any case, not suggested by the references or the art.

For all the reasons stated herein, the rejections for obviousness were improper, and appellants respectfully requests that the Examiner's rejections of the claims be reversed.

Date: March 8, 2007

Respectfully submitted,

CARY L. BATES, et al.

[illegible]

By________________________________

Roy W. Truelson, Attorney
Registration No. 34,265
(507) 202-8725 (Cell) (507) 289-6256 (Office)

From: IBM Corporation
Intellectual Property Law
Dept. 917, Bldg. 006-1
3605 Highway 52 North
Rochester, MN 55901

APPENDIX OF CLAIMS

1. A method of tracing the activity of an expression, said method comprising the machine-implemented steps of:

(a) receiving, from a user, a specification of a machine-implemented process in which a trigger expression is to be traced;

(b) receiving, from a user, a specification of the trigger expression to be traced in the machine-implemented process, said trigger expression representing a non-executable data value having a state;

(c) responsive to steps (a) and (b), monitoring execution of said machine-implemented process to detect occurrences of a plurality of references to a location in machine memory representing a state of said trigger expression, wherein each said occurrence of a reference to a location in machine memory representing a state of said trigger expression occurs as a result of executing said machine-implemented process;

(d) responsive to each detected occurrence of a reference to said location in machine memory representing a state of said trigger expression, storing the respective state of the trigger expression at the time of the respective detected occurrence of a reference to said location in machine memory representing a state of said trigger expression to create a history of said trigger expression within the machine-implemented process, said storing step being performed without interrupting the machine-implemented process; and

(e) restoring the state of the trigger expression when requested.

2. The method of claim 1, further comprising:
   (a) imposing a condition onto the trigger expression; and
   (b) storing the state of the trigger expression only when the condition is satisfied.

4. The method of claim 1, further comprising:
   (a) displaying the history such that the state of the trigger expression each time the trigger expression was active can be displayed separately.

5. The method of claim 1, wherein the trigger expression is one which results in an L value during the machine-implemented process.

7. The method of claim 1, wherein the reference to said location in machine memory representing a state of said trigger expression is a Read and/or a Write.

8. The method of claim 1, further comprising:
   (a) receiving, from a user, a specification of at least one attached expression;
   (b) responsive to each detected occurrence of a reference to said location in machine memory representing a state of said trigger expression, storing the respective state of the at least one attached expression at the time of the respective detected occurrence of a reference to said location in machine memory representing a state of said trigger expression, the states of the at least one attached expression being associated with said history of said trigger expression within the machine-implemented process; and
   (c) restoring the state of the at least one attached expression when requested.

9. The method of claim 1, wherein the machine-implemented process is a computer program.

10. The method of claim 1, as included in an object level trace program.

11. The method of claim 1, as included in a debug program.

12. A method of tracing the activity of an expression in an executing computer program, said method comprising the machine-implemented steps of:

(a) receiving, from a user, a specification of the computer program in which a trigger expression resulting in an L value during the execution of the computer program is to be traced;

(b) receiving, from a user a specification of the trigger expression and any optional attachment expressions to be traced in the computer program, said trigger expression representing a non-executable data value having a state;

(c) imposing a condition onto the trigger expression;

(d) responsive to steps (a) and (b), monitoring execution of said computer program to detect occurrences of a plurality of accesses to a location in memory containing a state representing said trigger expression, wherein each said occurrence of an access to a location in memory containing a state representing said trigger expression occurs as a result of executing said computer program;

(e) responsive to each detected occurrence of an access to said location in memory containing a value representing said trigger expression, if said condition is satisfied, then storing the respective state of the trigger expression and any optional attachment expressions at the time of the respective detected occurrence of an access to said location in memory containing a state representing the trigger expression to create a snapshot corresponding to the respective detected occurrence of an access to said location in memory, the step of storing being accomplished without interrupting the process;

(f) creating a profile of the trigger expression comprising storing each snapshot;

(g) displaying the profile such that each snapshot can be displayed separately; and

(h) restoring the state of each snapshot, when requested.

25. An article of manufacture, comprising a data storage medium tangibly embodying a program of machine readable instructions executable by an electronic processing apparatus to perform method steps for operating an electronic processing apparatus, said method steps comprising the steps of:

(a) initiating a user interface to exchange data input/output with a user and an electronic processing apparatus;

(b) requesting and receiving a trigger expression from a user, said trigger expression representing a non-executable data value having a state;

(c) requesting and receiving a program identification of a program in which the trigger expression is to be traced;

(d) causing the electronic processing apparatus to execute the identified program;

(e) monitoring execution of the identified program to detect occurrences of a plurality of references to a location in memory representing a state of said trigger expression, wherein each said occurrence of a reference to a location in memory representing a state of said trigger expression occurs as a result of executing the identified program;

(f) responsive to each detected occurrence of a reference to said location in memory representing a state of said trigger expression, storing the respective state of the trigger expression at the time of the respective detected occurrence of a reference to said location in memory representing a state of the trigger expression to create a corresponding respective snapshot, said snapshots forming a history of said trigger expression during execution of the identified program, said storing step being performed without interrupting or otherwise stopping execution of the identified program;

(g) maintaining the capability to restore each snapshot and display each snapshot to the user.

26. The article of manufacture of claim 25, further comprising:

(a) requesting the user to assign conditions to the trigger expression whereupon when the conditions are satisfied, a snapshot of the trigger expression is stored.

27. The article of manufacture of claim 25, further comprising:

(a) requesting the user to indicate attached expression whose states are also stored in a corresponding snapshot whenever a snapshot is stored for the trigger expression.

28. A digital data processing device, comprising:

(a) at least one processor;

(b) a memory functionally connected to said at least one processor;

(c) a first computer program executable by said at least one processor;

(d) at least one input device receiving input from a user and at least one output device for presenting output to a user;

(e) a second computer program for tracing said first computer program, said second computer program: (i) receiving a specification of a trigger expression used by said first computer program from said user using said at least one input device, said trigger expression representing a non-executable data value having a state; (ii) responsive to receiving a specification of a trigger expression, monitoring execution of said first computer program to detect occurrences of a plurality of references to a location in said memory representing a value of said trigger expression, wherein each said occurrence of a reference to a location in said memory representing a value of said trigger expression occurs as a result of executing said first computer program; (iii) responsive to each detected occurrence of a reference to said location in said memory representing a value of said trigger expression, storing a corresponding snapshot containing state data of said first computer program, said state data including said trigger expression, said storing step being performed without interrupting said first computer program; (iv) creating a history of said trigger expression during execution of said first program from said snapshots; and (v) presenting said history to said user using said at least one output device.

29. The digital data processing device of claim 28, wherein said second computer program further receives a specification of at least one condition for capturing said snapshot, and performs said step of storing a corresponding snapshot containing state data of said first computer program only if said at least one condition is satisfied.

30. The digital data processing device of claim 28, wherein said second computer program further received a specification of at least one attachment expression, and responsive thereto, includes said at least one attachment expression in said state data.

31. The digital data processing device of claim 28, wherein the first computer program and the second computer program execute on the same computer.

32. The digital data processing device of claim 28, wherein the first computer program and the second computer program execute on separate units connected by a data communications link.

APPENDIX OF EVIDENCE

No evidence is submitted.

APPENDIX OF RELATED PROCEEDINGS

There are no related proceedings.